Interoperability Land™

User Guide

Version 2 | May 2020

This guide is targeted at Developers

MiHIN
MICHIGAN HEALTH INFORMATION NETWORK
SHARED SERVICES

INTEROPERABILITY
INSTITUTE

Table of Contents

Overview

Interoperability Land™

Interoperability Land is a cloud-hosted digital collaboration platform designed to power the future of multi-organization development, integration, acceptance, and testing of innovative technologies and open standards. Organizations can learn, build and test healthcare applications and services with no risk of PHI disclosure using highly realistic, clinically relevant, synthetic patient data.
Two types of synthetic data are available in IOL:

- **PatientGen™** is an HL7 FHIR-compatible test data generator that models a simulated healthcare network of providers, practices, and hospitals with a large population of patients who experience changes in their health status and mortality risks for many important medical conditions and procedures.
- **Personas** are realistic and complete synthetic representations of a person. Personas have unique attitudes, conditions, and environments that affect how they interact with each other and the healthcare system.

Both are designed to showcase new technology, promote standards (e.g. HL7 FHIR®), and accelerate interoperability.
This synthetic ecosystem allows healthcare organizations to:

- Demonstrate apps and services in an engaging and meaningful way using data visualization to reveal interoperability between systems
- Rigorously test and certify that applications meet standards, performance and scalability requirements
- Collaborate with other organizations to develop and test interoperable, standards-compliant solutions
- Host collaborative events to promote learning and standards-based technology adoption
- Deliver higher quality applications and services faster to market

Terminology

InterOp PIT: Pilot Interoperability Testbed, a Fast Healthcare Interoperability Resources (FHIR®) server that represents a real-world healthcare organization, populated with synthetic patient data.
Patient-Gen™: A synthetic patient data generator.
Persona: A hand-made synthetic patient, custom fitted to align with various use-cases.
IOL Ring: A grouping of various servers that includes EHRs, Pharmacies, Payers, and an HIE, populated with synthetic data that is choreographed across each to emulate a real-world healthcare ecosystem.
HL7 FHIR: Health Level Seven's Fast Healthcare Interoperability Resources
Resource: Any data file containing synthetic patient data. Formatted in JSON.
Sandbox: The development environment that contains all PITs and IOL Ring
HAPI FHIR®: A user interface (UI) that assists in interacting with a FHIR server through querying and displaying FHIR data.

Introduction to FHIR

Healthcare Interoperability Resources (FHIR)

Health Level Seven International (HL7), a not-for-profit organization that develops and standardizes data framework for the exchange of electronic health information, has developed a specification standard known as Fast Healthcare Interoperability Resources (FHIR).

Why FHIR Is Important

FHIR is designed to help health information organizations more quickly and easily exchange and retrieve data from electronic health record (EHR) systems, and to help health IT developers more efficiently build applications to support this exchange of information.

How FHIR Works and Understanding FHIR Resources

FHIR frameworks are built around the concept of “resources” – these objects are basic, modular units of interoperability that can be assembled into working systems to try to resolve clinical, administrative and infrastructural problems in healthcare.
Administrative concepts such as patients, providers, organizations and devices, as well as a variety of clinical concepts including conditions, medications, diagnostics, care plans and claims information, among others are translated into FHIR resources with structured and standardized data for easy interoperability between EHR vendors and other software development resources and tools.
FHIR is designed specifically for the web and provides resources and foundations in two formats XML and JSON.
More information on FHIR can be found at: https://www.hl7.org/fhir/overview.html

Other FHIR Initiatives

SMART on FHIR has also gained broad industry support. The SMART on FHIR initiative is based at Boston Children’s Hospital and features a set of open specifications to integrate apps with EHRs, portals, health information exchanges and other health IT systems.
Another initiative is HAPI FHIR (“happy fire”), an open source Java implementation of the FHIR specification. It was developed at University Health Network in Ontario, Canada. HAPI FHIR is free to use.

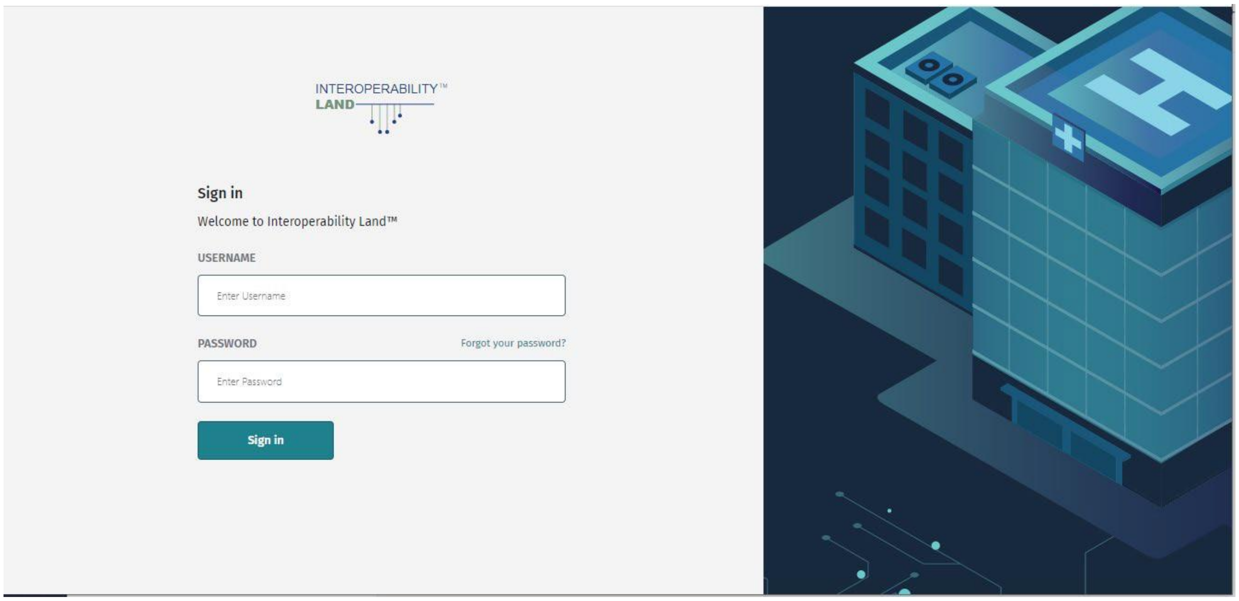
Getting Started with IOL

Getting Access

Participants in an event will receive a temporary login sent to the email address used to register for the event. Click the link in the email, or open your internet browser and enter the following URL: https://www.interopland.com/login

Logging In

Log into Interoperability Land using the provided temporary credentials. Upon logging in you will be prompted to enter a new password and accept the terms and conditions.



Sign in

Welcome to Interoperability Land™

USERNAME

Enter Username

PASSWORD

Forgot your password?

Enter Password

Sign in

Exploring the Homepage & Site Navigation

Homepage

The image below serves as the home page for IOL and can be accessed at any time by clicking the Interoperability Land logo in the top-left corner of your browser.

Navigation Menu

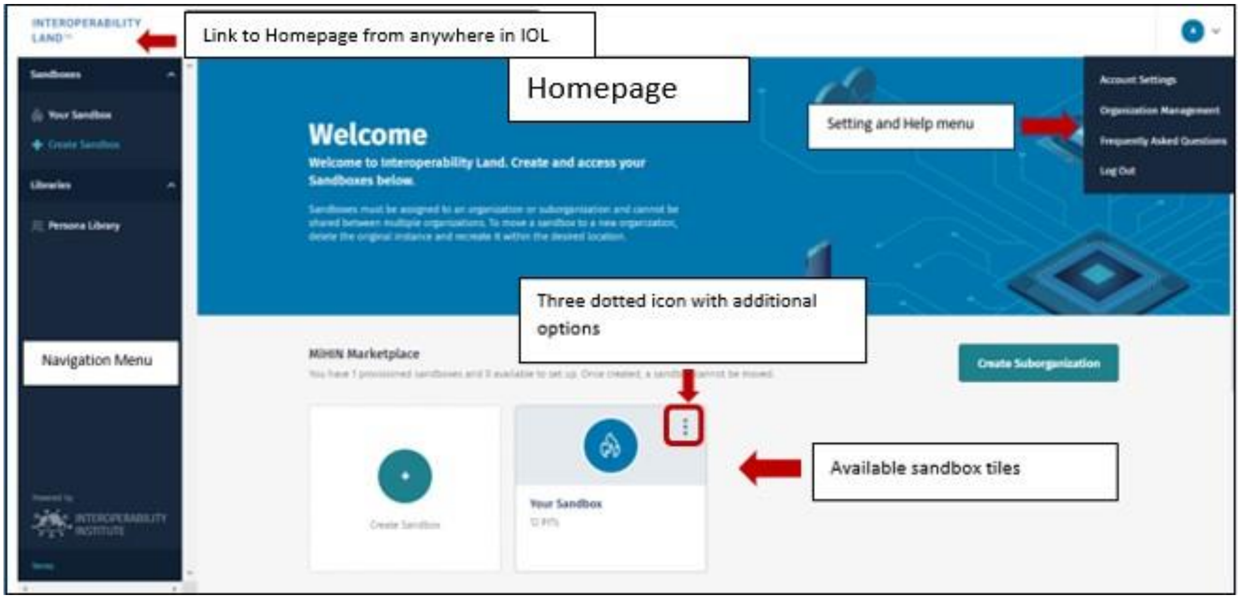
On the left-hand side of the browser you will see the navigation menu. This menu contains links to all **Sandboxes** and the **Persona library** you have access to.

Settings and Help Menu

In the top-right corner of the browser, you will see your profile icon. Clicking this opens a drop-down menu containing **Account Settings, Organization Management, Frequently Asked Questions,** and **Log Out**.

Dashboard

In the main section of the dashboard you can find your organizations, sub-organizations, and Sandboxes. The tiles on your homepage include a three-dotted icon with additional options.



Welcome to Interoperability Land. Create and access your Sandboxes below.

Sandboxes must be assigned to an organization or suborganization and cannot be shared between multiple organizations. To move a sandbox to a new organization, delete the original instance and recreate it within the desired location.

You have 1 provisioned sandboxes and 3 available to set up. Once created, a sandbox cannot be moved.

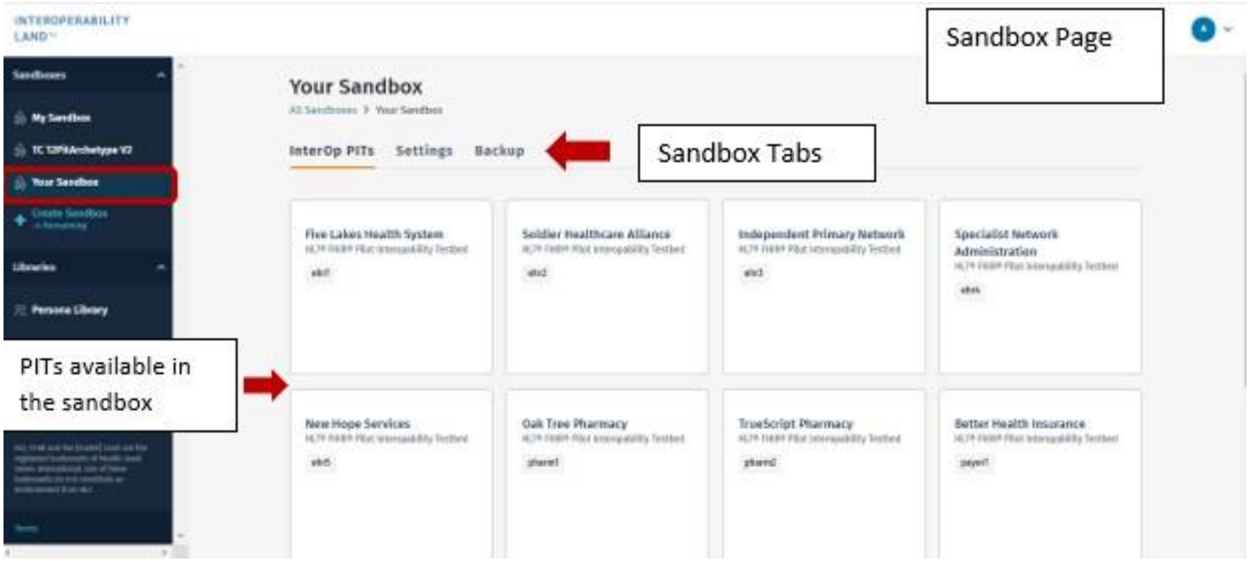
Exploring the Sandbox

Accessing your Sandbox

You can access your sandbox by clicking either the tile on the homepage or the link in the navigation menu side-bar.

Sandbox Tab Menu

The tab menu at the top allows you to cycle through the Interop PITs display, Sandbox Settings, and Sandbox Backup. In the settings tab, you may delete your Sandbox. In the Backup tab, you can create up to **2** backups of your Sandbox or **Reset to Initial State**. **Note:** Resetting to initial state does **not** clear your backups.


INTEROPERABILITY LAND™
Sandboxes
My Sandbox
TC 12PIArchetype V2
Your Sandbox
Create Sandbox
Libraries
Persona Library
Your Sandbox
All Sandboxes > Your Sandbox
InterOp PITs
Settings
Backup
Five Lakes Health System
HL7® FHIR® Pilot Interoperability Testbed
ehr1
Soldier Healthcare Alliance
HL7® FHIR® Pilot Interoperability Testbed
ehr2
Independent Primary Network
HL7® FHIR® Pilot Interoperability Testbed
ehr3
Specialist Network Administration
HL7® FHIR® Pilot Interoperability Testbed
ehr4
New Hope Services
HL7® FHIR® Pilot Interoperability Testbed
ehr5
Oak Tree Pharmacy
HL7® FHIR® Pilot Interoperability Testbed
pharm1
TrueScript Pharmacy
HL7® FHIR® Pilot Interoperability Testbed
pharm2
Better Health Insurance
HL7® FHIR® Pilot Interoperability Testbed
payer1
Terms

Sandbox Page

Sandbox Tabs

PITs available in the sandbox

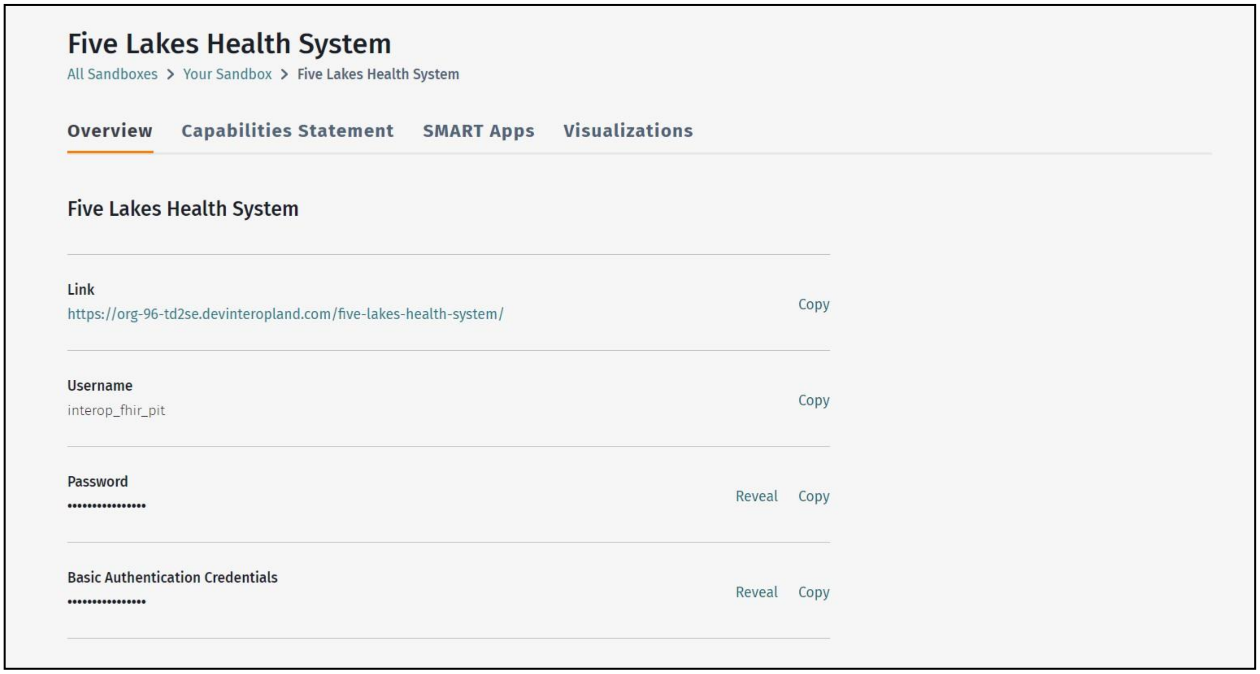
Exploring the PITs

InterOp PIT Tiles

Each tile under the InterOp PITs tab contains basic information about the PIT and is also a button to access its details menu. Simply click a tile to access more details about the PIT.

PIT Tab Menu

The Tab menu under the PIT contains the Overview, Capabilities Statement, SMART Apps, and Visualizations. The overview contains a link to access the PIT, along with the PIT's username, password, and authentication credentials.



Five Lakes Health System

All Sandboxes > Your Sandbox > Five Lakes Health System

Five Lakes Health System

Link
https://org-96-td2se.devinteropland.com/five-lakes-health-system/

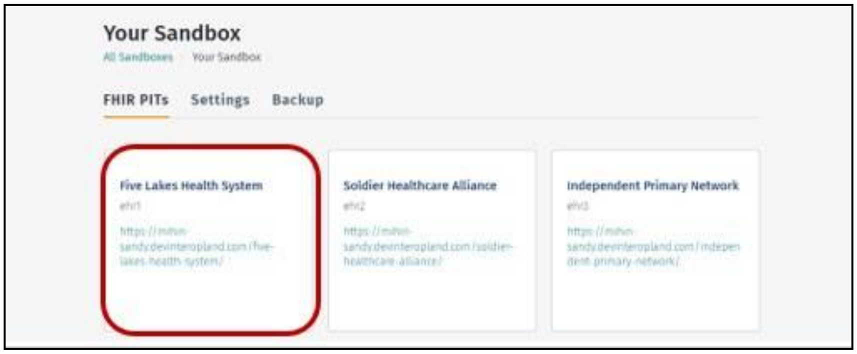
Username
interop_fhir_pit

Password
•••••••••••••••

Basic Authentication Credentials
•••••••••••••••

Connecting a SMART on FHIR App

Select the PIT you would like to connect to the SMART of FHIR App.



Your Sandbox

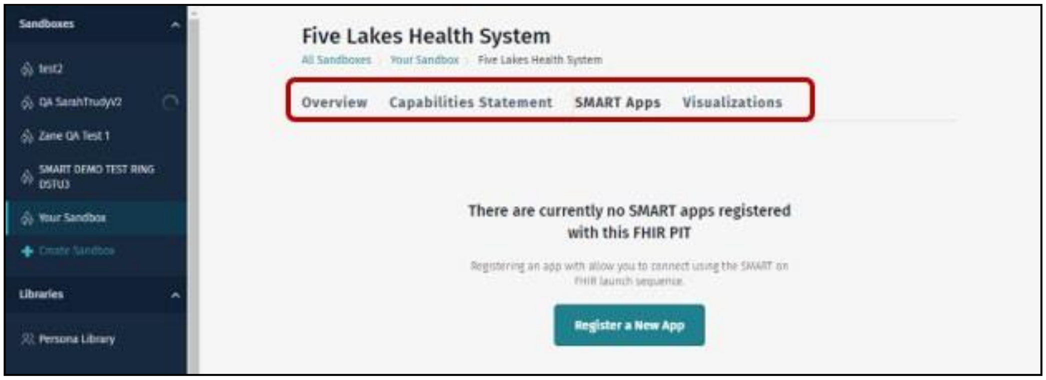
All Sandboxes Your Sandbox

Five Lakes Health System
ehr1
https://mihin-sandy.devinteropland.com/five-lakes-health-system/

Soldier Healthcare Alliance
ehr2
https://mihin-sandy.devinteropland.com/soldier-healthcare-alliance/

Independent Primary Network
ehr3
https://mihin-sandy.devinteropland.com/independent-primary-network/

Click the SMART Apps tab within the PIT. Then click Register a New App.



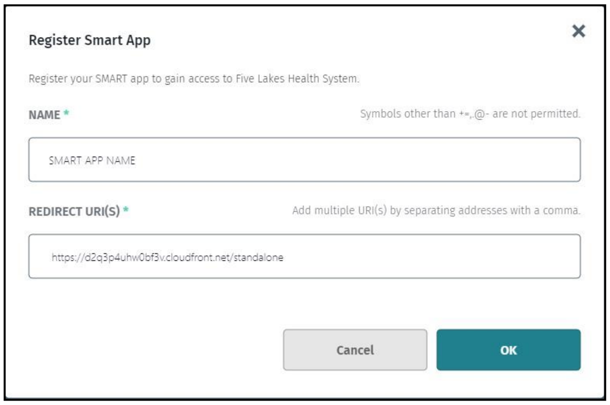
Five Lakes Health System

All Sandboxes Your Sandbox Five Lakes Health System

There are currently no SMART apps registered with this FHIR PIT

Registering an app with allow you to connect using the SMART on FHIR launch sequence.

Enter a name for the app along with the redirect URL.



Register Smart App

Register your SMART app to gain access to Five Lakes Health System.

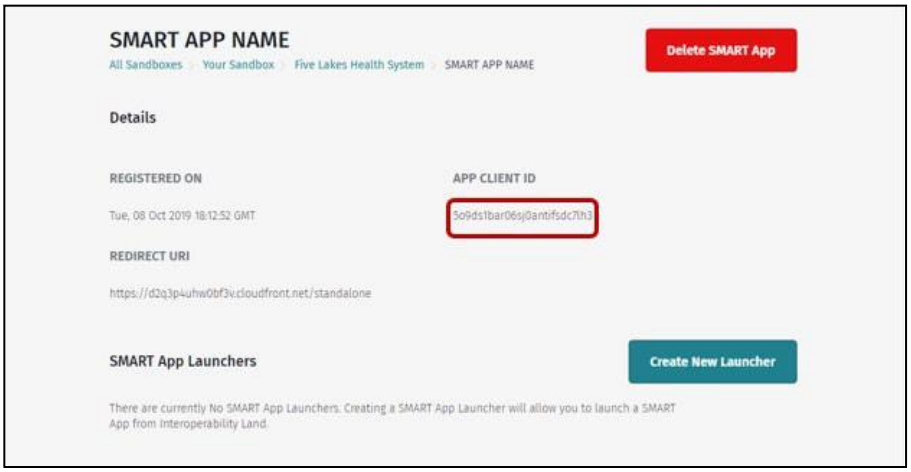
NAME * — Symbols other than +=,.@- are not permitted.

SMART APP NAME

REDIRECT URI(S) * — Add multiple URI(s) by separating addresses with a comma.

https://d2q3p4uhw0bf3v.cloudfront.net/standalone

After clicking "OK" the details will appear, including the App Client ID that is used to connect from the App.



SMART APP NAME

All Sandboxes › Your Sandbox › Five Lakes Health System › SMART APP NAME

Details

REGISTERED ON

Tue, 08 Oct 2019 18:12:52 GMT

APP CLIENT ID

5o9ds1bar06sj0antifsdc7lh3

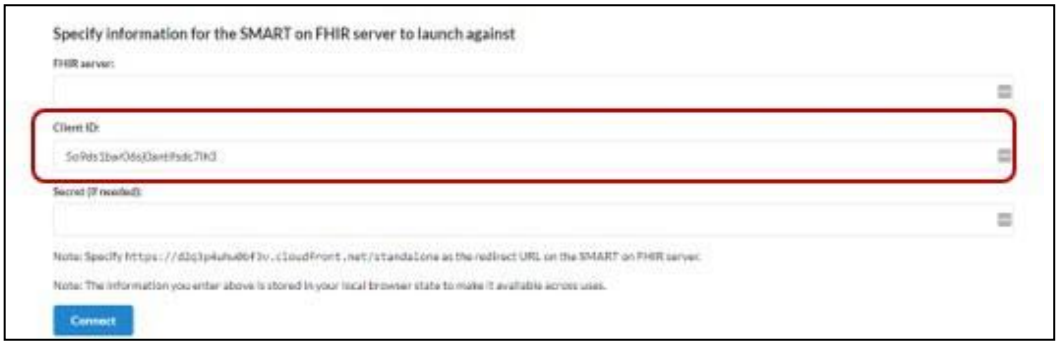
REDIRECT URI

https://d2q3p4uhw0bf3v.cloudfront.net/standalone

SMART App Launchers

There are currently No SMART App Launchers. Creating a SMART App Launcher will allow you to launch a SMART App from Interoperability Land.

Enter the App client ID on the **app side**. Example:



Specify information for the SMART on FHIR server to launch against

FHIR server:

Client ID:

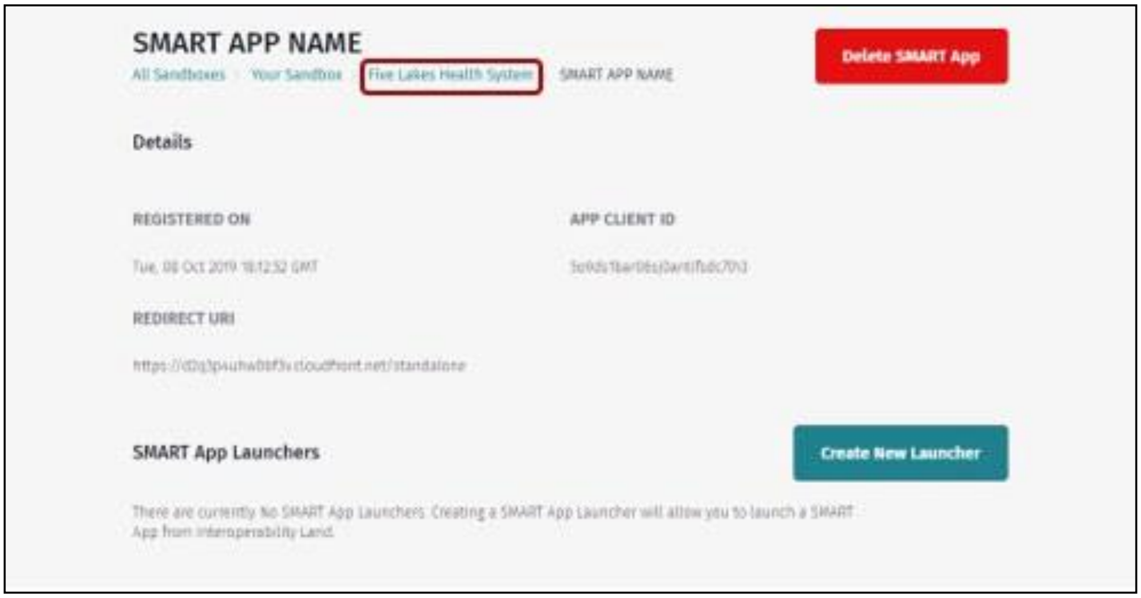
5o9ds1bar06sj0antifsdc7lh3

Secret (if needed):

Note: Specify https://d2q3p4uhw0bf3v.cloudfront.net/standalone as the redirect URL on the SMART on FHIR server.

Note: The information you enter above is stored in your local browser state to make it available across uses.

Navigate to the PIT via the breadcrumb or the Sandbox link in the Navigation Menu.



SMART APP NAME

All Sandboxes › Your Sandbox › Five Lakes Health System › SMART APP NAME

Details

REGISTERED ON

Tue, 08 Oct 2019 18:12:52 GMT

APP CLIENT ID

5o9ds1bar06sj0artifbdc7013

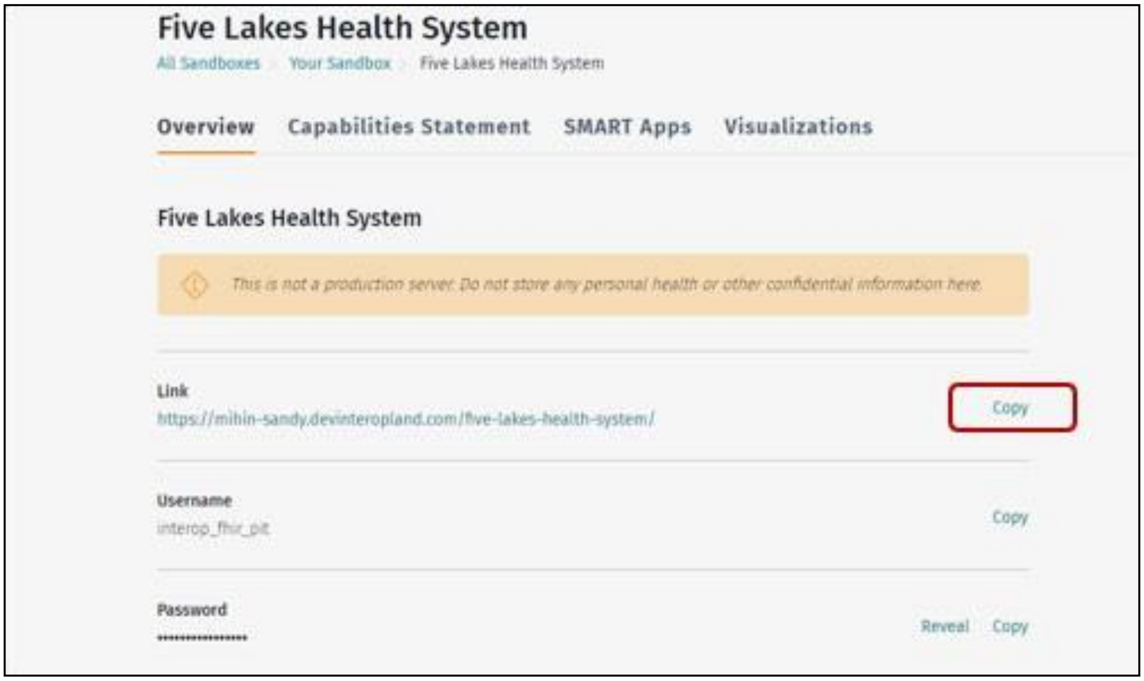
REDIRECT URI

https://d2q3p4uhwb8f3v.cloudfront.net/standalone

SMART App Launchers

There are currently No SMART App Launchers. Creating a SMART App Launcher will allow you to launch a SMART App from Interoperability Land.

Copy the URL address link on the PIT's Overview tab.



Five Lakes Health System

All Sandboxes › Your Sandbox › Five Lakes Health System

Five Lakes Health System

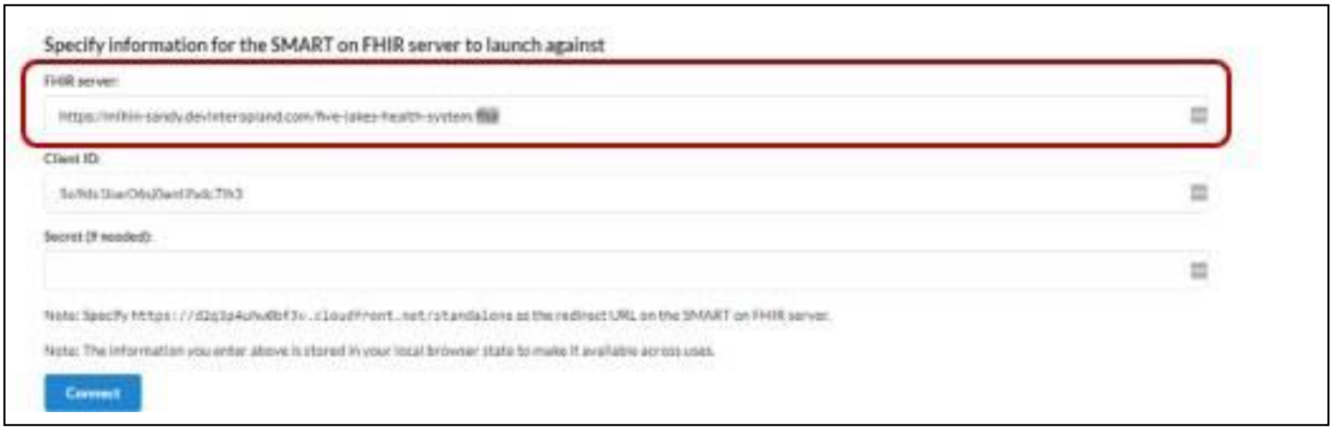
This is not a production server. Do not store any personal health or other confidential information here.

Link
https://mihin-sandy.devinteropland.com/five-lakes-health-system/

Username
interop_fhir_pit

Password
••••••••••••••••

Paste the URL address as the FHIR Server address on the app side, then add **/fhir** at the end of the URL. Example: https://organization.interopland.com/five-lakes-health-system/**fhir**



Specify information for the SMART on FHIR server to launch against

FHIR server:

https://mihin-sandy.dev.interopland.com/five-lakes-health-system/fhir

Client ID:

[illegible]

Secret (if needed):

Note: Specify https://d2q3p4uhw0bf3v.cloudfront.net/standalone as the redirect URL on the SMART on FHIR server.

Note: The information you enter above is stored in your local browser state to make it available across uses.

Select “Connect” and your SMART on FHIR App will launch.

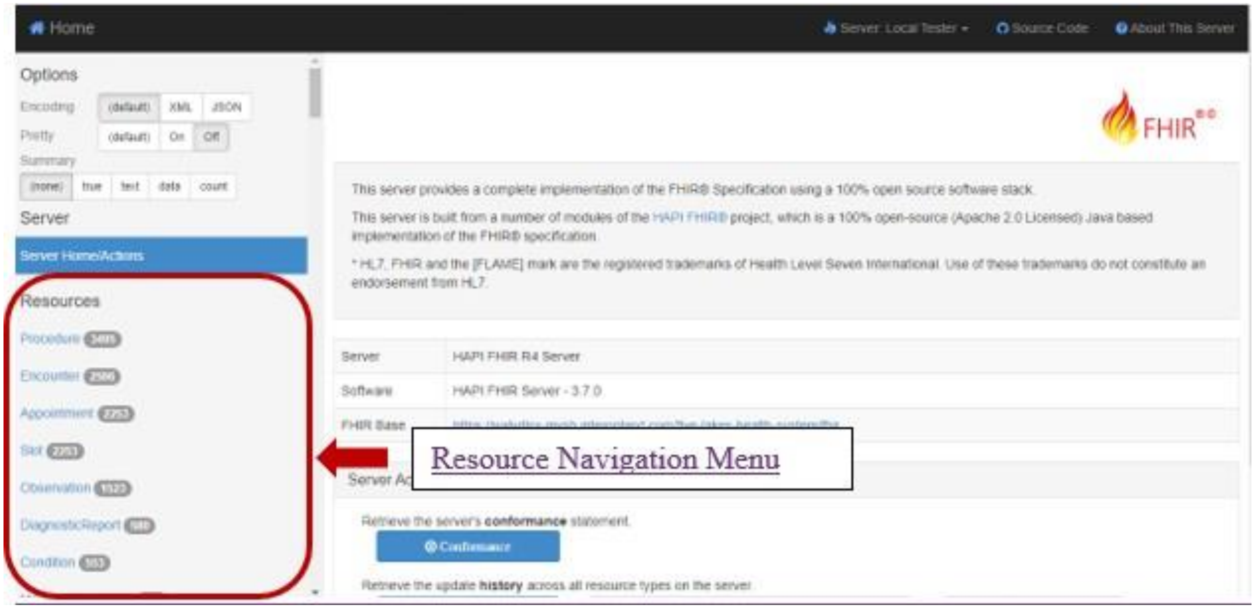
Completing a Query

Accessing the PIT and HAPI FHIR API

Follow the link located under the Overview tab within the PIT you are trying to query. This will access the HAPI FHIR API’s user interface.

Navigating the Resources

The FHIR Resources are displayed in the left-hand navigation menu alongside the number of instances of each resource.



This server provides a complete implementation of the FHIR® Specification using a 100% open source software stack.

This server is built from a number of modules of the HAPI FHIR® project, which is a 100% open-source (Apache 2.0 Licensed) Java based implementation of the FHIR® specification.

* HL7, FHIR and the [FLAME] mark are the registered trademarks of Health Level Seven International. Use of these trademarks do not constitute an endorsement from HL7.

Server	HAPI FHIR R4 Server
Software	HAPI FHIR Server - 3.7.0
FHIR Base	[illegible]

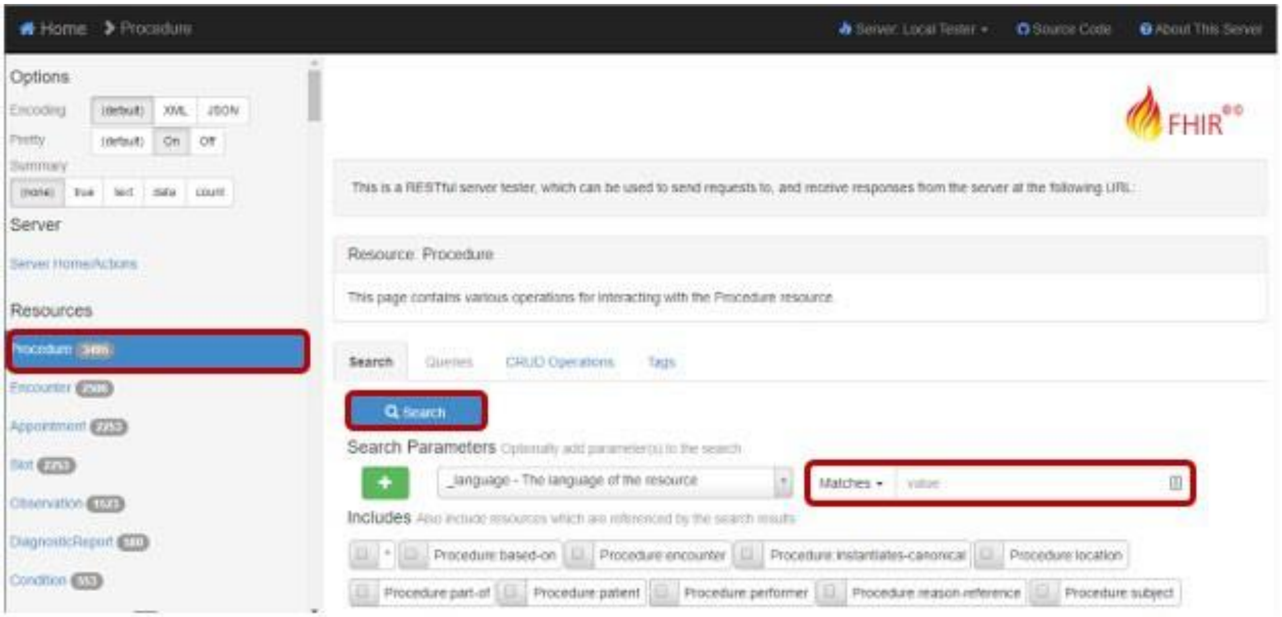
Server Ac

Retrieve the server's **conformance** statement.

Retrieve the update **history** across all resource types on the server.

Performing a Basic Query

Find the type of resource you would like to query in the resource navigation menu and select it. A search tool will appear with several options to modify your query. To perform a basic query and pull every resource of the type you selected, leave the values at default/blank then click "search."



This is a RESTful server tester, which can be used to send requests to, and receive responses from the server at the following URL:

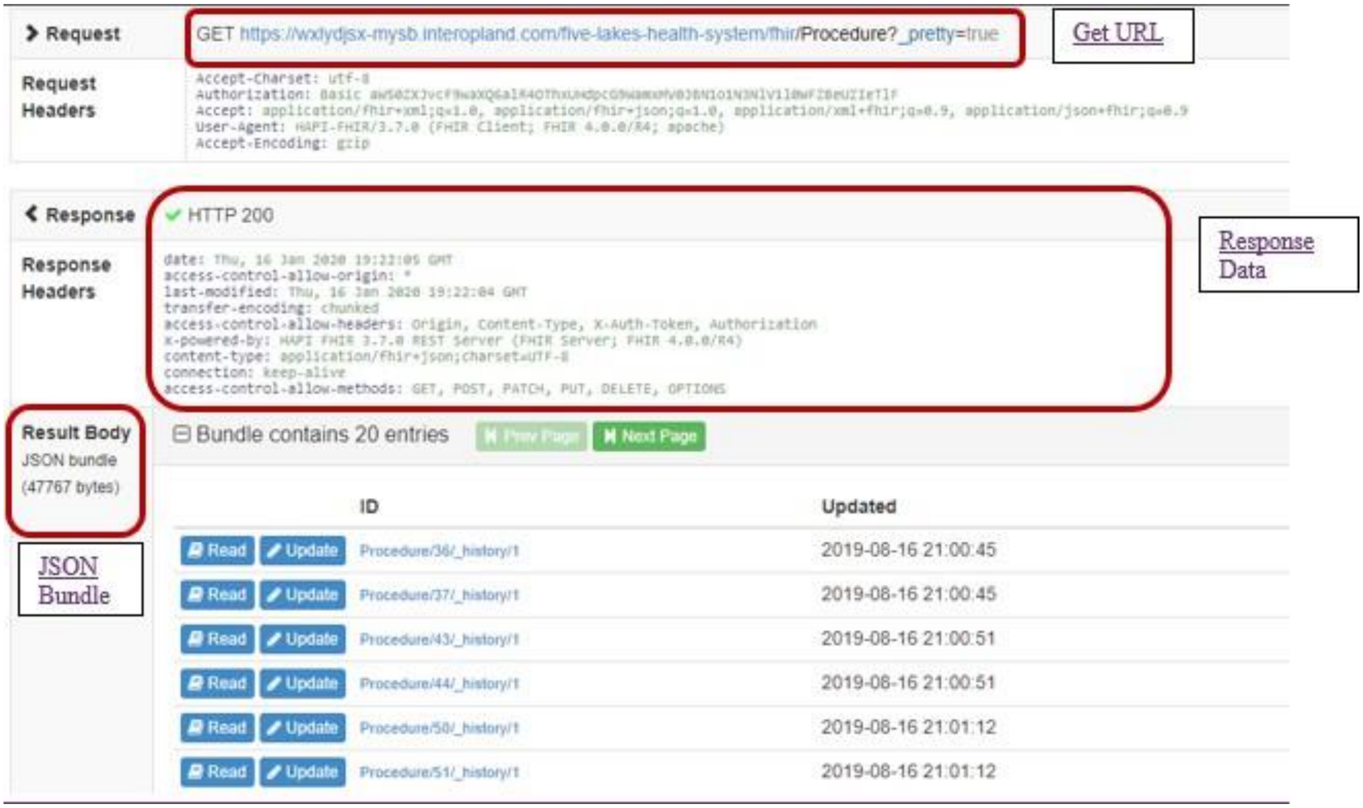
Resource: Procedure

This page contains various operations for interacting with the Procedure resource.

Search Parameters Optionally add parameter(s) to the search

Includes Also include resources which are referenced by the search results

Within the result body section, you will find a GET URL, Response data, and a JSON bundle in both individual files and raw bundle formats (see screenshots below).



Request GET https://wxlydjsx-mysb.interopland.com/five-lakes-health-system/fhir/Procedure?_pretty=true

Request Headers

```
Accept-Charset: utf-8
Authorization: Basic aW50ZXJvcF9waXQ6alR4OThxU4dpcG9WamxMV0JBN1o1N3NlV1l0WFZBeUZIeTlF
Accept: application/fhir+xml;q=1.0, application/fhir+json;q=1.0, application/xml+fhir;q=0.9, application/json+fhir;q=0.9
User-Agent: HAPI-FHIR/3.7.0 (FHIR Client; FHIR 4.0.0/R4; apache)
Accept-Encoding: gzip
```

Response ✓ HTTP 200

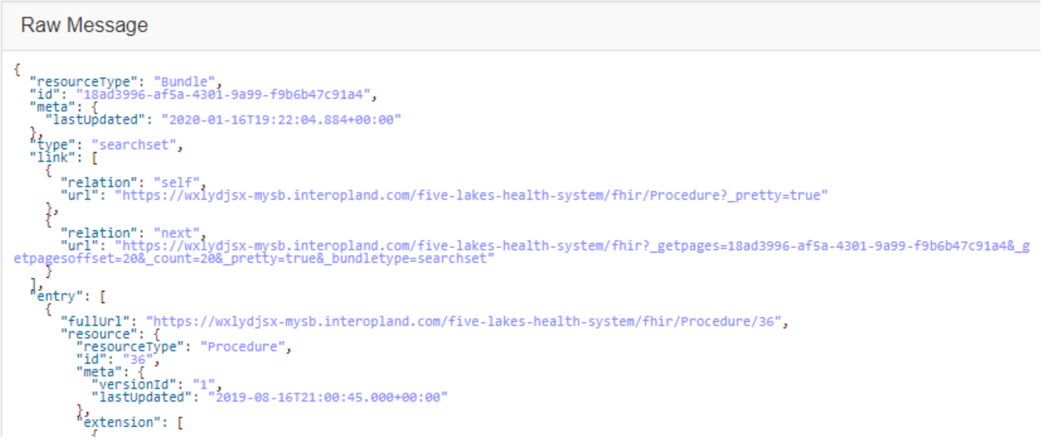
Response Headers

```
date: Thu, 16 Jan 2020 19:22:05 GMT
access-control-allow-origin: *
last-modified: Thu, 16 Jan 2020 19:22:04 GMT
transfer-encoding: chunked
access-control-allow-headers: Origin, Content-Type, X-Auth-Token, Authorization
x-powered-by: HAPI FHIR 3.7.0 REST Server (FHIR Server; FHIR 4.0.0/R4)
content-type: application/fhir+json;charset=UTF-8
connection: keep-alive
access-control-allow-methods: GET, POST, PATCH, PUT, DELETE, OPTIONS
```

Result Body JSON bundle (47767 bytes)

⊟ Bundle contains 20 entries Prev Page Next Page

	ID	Updated
Read Update	Procedure/36/_history/1	2019-08-16 21:00:45
Read Update	Procedure/37/_history/1	2019-08-16 21:00:45
Read Update	Procedure/43/_history/1	2019-08-16 21:00:51
Read Update	Procedure/44/_history/1	2019-08-16 21:00:51
Read Update	Procedure/50/_history/1	2019-08-16 21:01:12
Read Update	Procedure/51/_history/1	2019-08-16 21:01:12



Raw Message

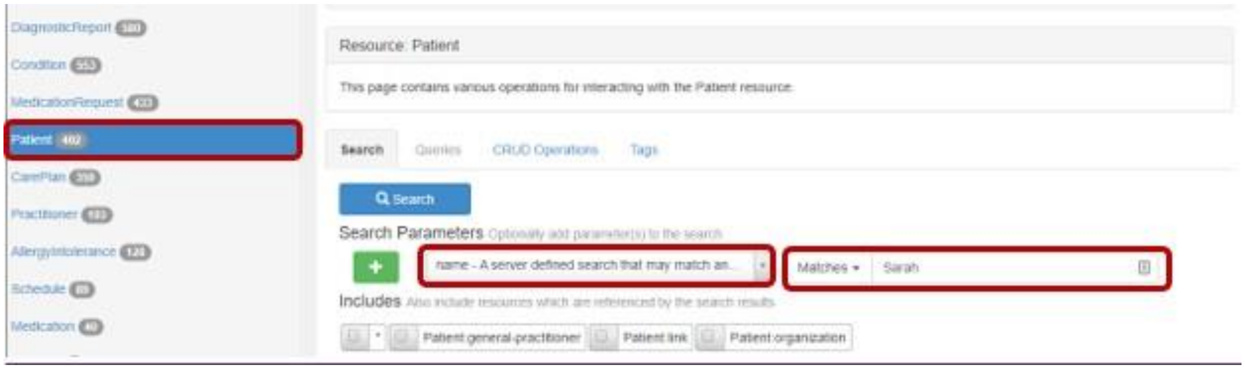
```
{
  "resourceType": "Bundle",
  "id": "18ad3996-af5a-4301-9a99-f9b6b47c91a4",
  "meta": {
    "lastUpdated": "2020-01-16T19:22:04.884+00:00"
  },
  "type": "searchset",
  "link": [
    {
      "relation": "self",
      "url": "https://wxlydjsx-mysb.interopland.com/five-lakes-health-system/fhir/Procedure?_pretty=true"
    },
    {
      "relation": "next",
      "url": "https://wxlydjsx-mysb.interopland.com/five-lakes-health-system/fhir?_getpages=18ad3996-af5a-4301-9a99-f9b6b47c91a4&_g
etpagesoffset=20&_count=20&_pretty=true&_bundletype=searchset"
    }
  ],
  "entry": [
    {
      "fullUrl": "https://wxlydjsx-mysb.interopland.com/five-lakes-health-system/fhir/Procedure/36",
      "resource": {
        "resourceType": "Procedure",
        "id": "36",
        "meta": {
          "versionId": "1",
          "lastUpdated": "2019-08-16T21:00:45.000+00:00"
        },
        "extension": [
```

Performing an Advanced Query

In the resource’s search tool, click the drop-down next to the green plus button to select a datapoint within the resource to query. Then enter a value to query. The server will return the same data type as before: GET URL, Response data, and a JSON bundle in both individual files and raw bundle formats.

Searches for a patient can be done with at least one identifying piece of information such as name, social security number or patient ID.

Example: Click the **Patient** resource, then switch the search parameter to **name**, and enter the value **Sarah**. Click search.



Resource: Patient

This page contains various operations for interacting with the Patient resource.

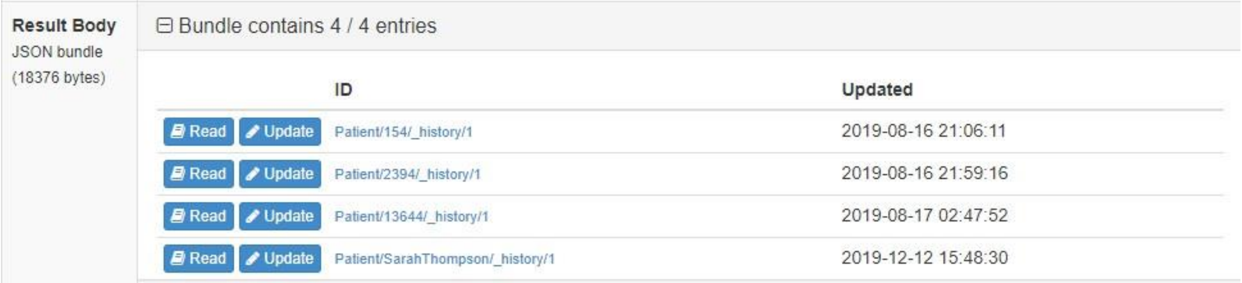
Search Parameters Optionally add parameter(s) to the search

name - A server defined search that may match an... | Matches | Sarah

Includes Also include resources which are referenced by the search results

□ * □ Patient:general-practitioner □ Patient:link □ Patient:organization

This results in a bundle of Patient resources that all contain Sarah within the name value array (as highlighted in screen shot below).



Result Body
JSON bundle
(18376 bytes)

⊟ Bundle contains 4 / 4 entries

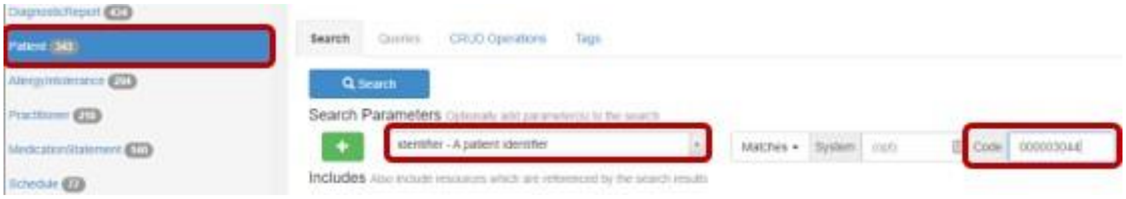
	ID	Updated
Read Update	Patient/154/_history/1	2019-08-16 21:06:11
Read Update	Patient/2394/_history/1	2019-08-16 21:59:16
Read Update	Patient/13644/_history/1	2019-08-17 02:47:52
Read Update	Patient/SarahThompson/_history/1	2019-12-12 15:48:30

Queries - Using the HAPI FHIR Interface

Query 1 – Finding A Patient By SSN

Because the HAPI FHIR Interface only supports a limited set of search parameters, an additional initial query is necessary to find the Patient record by SSN. The id of the Patient record we find will be used in future queries.

Searching for a patient requires selecting the Patient option from the resource's navigation menu on the left side of the page. Specifying the SSN to search for is then accomplished by selecting the identifier option in the search parameters dropdown, then entering the target SSN in the code field to the right (as seen below SSN 000003044 is used in this example).

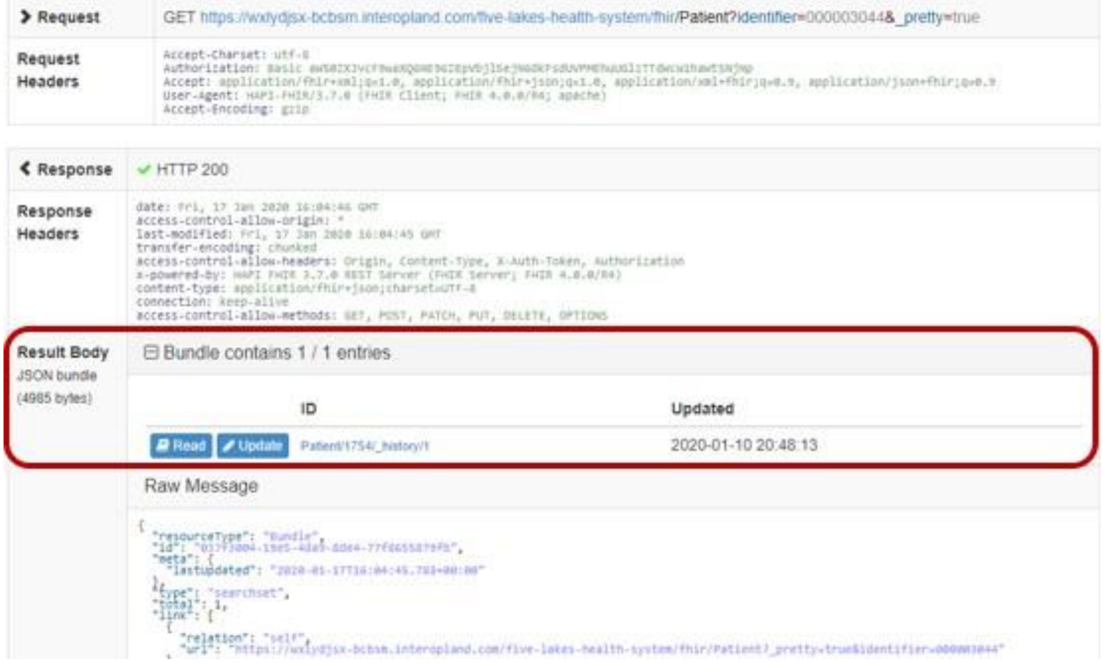


Search Parameters Optionally add parameter(s) to the search

+ identifier - A patient identifier | Matches ▾ | System (opt) | Code 000003044

Includes Also include resources which are referenced by the search results

This query produces the result shown below. Pay particular attention to the lower-right area of the screen, which contains the section labelled Result Body. This is the data returned in response to the query.



> Request GET https://wxlydjsx-bcbsm.interopland.com/five-lakes-health-system/fhir/Patient?identifier=000003044&_pretty=true

Request Headers

```
Accept-Charset: utf-8
Authorization: Basic [illegible]
Accept: application/fhir+xml;q=1.0, application/fhir+json;q=1.0, application/xml+fhir;q=0.9, application/json+fhir;q=0.9
User-Agent: HAPI-FHIR/3.7.0 (FHIR Client; FHIR 4.0.0/R4; apache)
Accept-Encoding: gzip
```

< Response ✓ HTTP 200

Response Headers

```
date: Fri, 17 Jan 2020 16:04:46 GMT
access-control-allow-origin: *
last-modified: Fri, 17 Jan 2020 16:04:45 GMT
transfer-encoding: chunked
access-control-allow-headers: Origin, Content-Type, X-Auth-Token, Authorization
x-powered-by: HAPI FHIR 3.7.0 REST Server (FHIR Server; FHIR 4.0.0/R4)
content-type: application/fhir+json;charset=UTF-8
connection: keep-alive
access-control-allow-methods: GET, POST, PATCH, PUT, DELETE, OPTIONS
```

Result Body
JSON bundle
(4985 bytes)

⊟ Bundle contains 1 / 1 entries

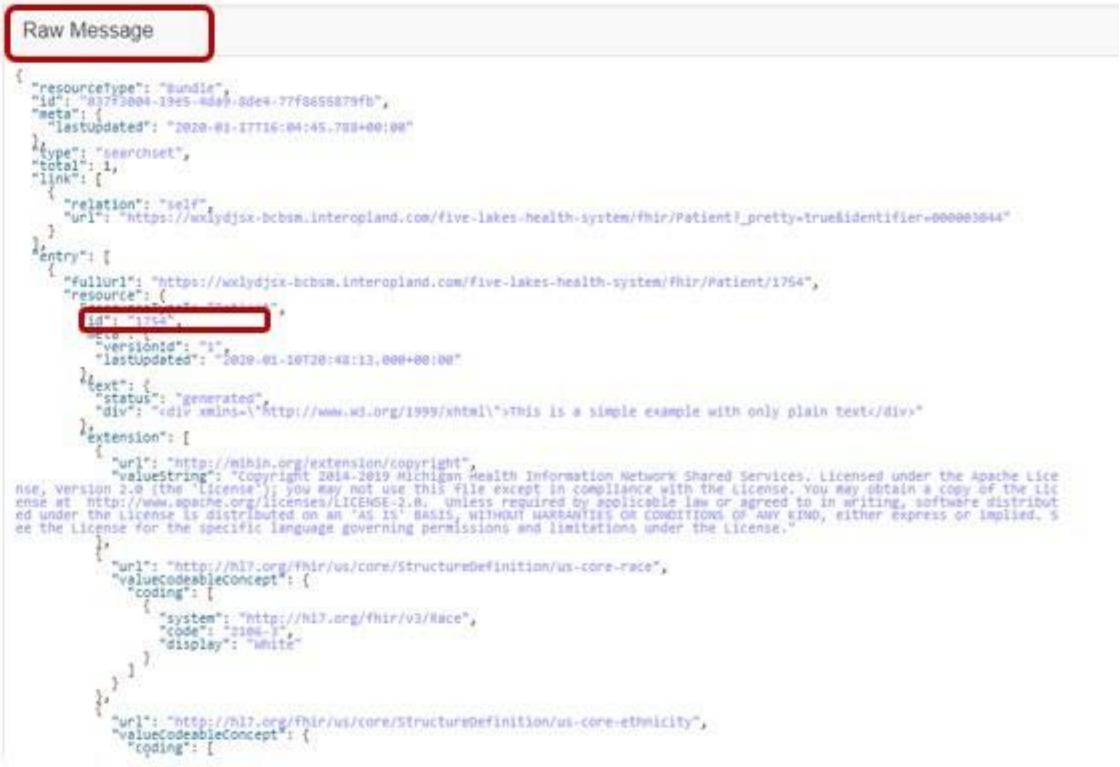
	ID	Updated
Read Update	Patient/1754/_history/1	2020-01-10 20:48:13

Raw Message

```
{
  "resourceType": "Bundle",
  "id": "[illegible]-1be5-4da5-8de4-77fd65587[illegible]",
  "meta": {
    "lastUpdated": "2020-01-17T16:04:45.783+00:00"
  },
  "type": "searchset",
  "total": 1,
  "link": [
    {
      "relation": "self",
      "url": "https://wxlydjsx-bcbsm.interopland.com/five-lakes-health-system/fhir/Patient?_pretty=true&identifier=000003044"
```

The Result Body section in this case contains a line at the top that reads “Bundle contains 1 / 1 entries”. This line shows how many results matched the query that was specified; the first number is the count of results included in this response, the second is the total number of records satisfying that query in this PIT. When a query matches a large number of records, the numbers can be different because the FHIR server has a limit on the number of records it can return in a single response.

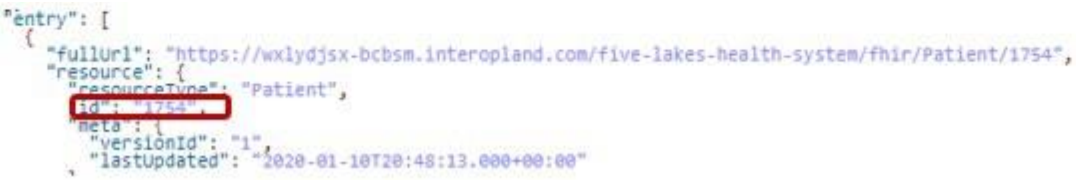
Below the summary section is a section labelled Raw Message. This section contains the actual data sought by the query (in this case, a Patient record identified by the supplied SSN).



Raw Message

```
{
  "resourceType": "Bundle",
  "id": "837f3004-19e5-4da9-8de4-77f8655879fb",
  "meta": {
    "lastUpdated": "2020-01-17T16:04:45.788+00:00"
  },
  "type": "searchset",
  "total": 1,
  "link": [
    {
      "relation": "self",
      "url": "https://wxlydjsx-bcbsm.interopland.com/five-lakes-health-system/fhir/Patient?_pretty=true&identifier=000003044"
    }
  ],
  "entry": [
    {
      "fullUrl": "https://wxlydjsx-bcbsm.interopland.com/five-lakes-health-system/fhir/Patient/1754",
      "resource": {
        "resourceType": "Patient",
        "id": "1754",
        "meta": {
          "versionId": "1",
          "lastUpdated": "2020-01-10T20:48:13.000+00:00"
        },
        "text": {
          "status": "generated",
          "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">This is a simple example with only plain text</div>"
        },
        "extension": [
          {
            "url": "http://mihin.org/extension/copyright",
            "valueString": "Copyright 2014-2019 Michigan Health Information Network Shared Services. Licensed under the Apache Lice
nse, Version 2.0 (the 'License'); you may not use this file except in compliance with the License. You may obtain a copy of the Lic
ense at  http://www.apache.org/licenses/LICENSE-2.0.  Unless required by applicable law or agreed to in writing, software distribut
ed under the License is distributed on an 'AS IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. S
ee the License for the specific language governing permissions and limitations under the License."
          },
          {
            "url": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-race",
            "valueCodeableConcept": {
              "coding": [
                {
                  "system": "http://hl7.org/fhir/v3/Race",
                  "code": "2106-3",
                  "display": "White"
                }
              ]
            }
          },
          {
            "url": "http://hl7.org/fhir/us/core/StructureDefinition/us-core-ethnicity",
            "valueCodeableConcept": {
              "coding": [
```

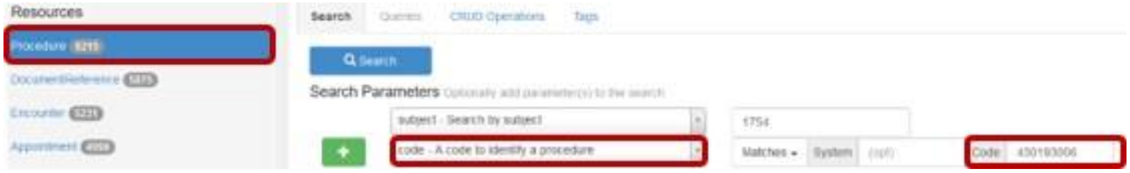
There is a wealth of information about the patient here, but the main field of interest for purposes of this exercise is the id. Specified under the “resource” element of a member of the “entry” collection, this is the unique identifier of this particular patient record within the PIT. It is what other records will use to refer to this patient and thus can be used to search for only records relevant to this patient.



```
"entry": [
  {
    "fullUrl": "https://wxlydjsx-bcbsm.interopland.com/five-lakes-health-system/fhir/Patient/1754",
    "resource": {
      "resourceType": "Patient",
      "id": "1754",
      "meta": {
        "versionId": "1",
        "lastUpdated": "2020-01-10T20:48:13.000+00:00"
      },
```

Query 2 – Finding Medication Reconciliation Procedures By Patient

When a Medication Reconciliation is performed, it is recorded under the Procedure resource. This resource can be queried using the subject parameter in the same fashion as the DocumentReference resource in the previous two queries. This will restrict the result to only procedures performed on   the specified patient. The search results can be further refined by using the code search parameter to retrieve only Medication Reconciliation procedures (based on the Snomed code for that procedure: 430193006).



Search Parameters Optionally add parameter(s) to the search

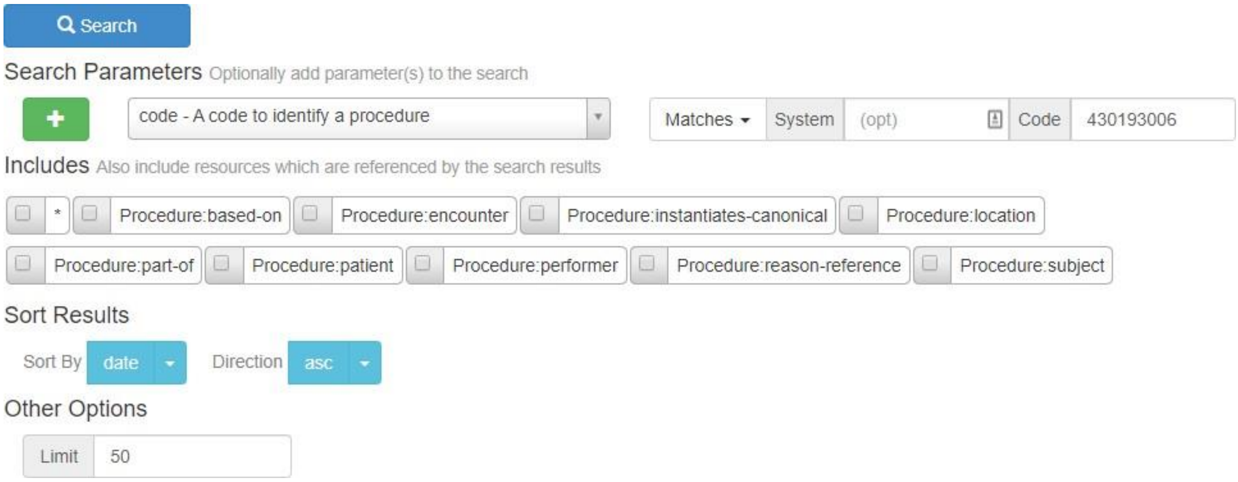
subject - Search by subject: 1754

code - A code to identify a procedure: Code 430193006

The result details each procedure that fits the specified criteria.

Query 3 – All Medication Reconciliation Procedures

Querying for all Medication Reconciliation procedures follows the same basic form as the DocumentReference bulk queries.



Search Parameters Optionally add parameter(s) to the search

code - A code to identify a procedure | Matches | System (opt) | Code 430193006

Includes Also include resources which are referenced by the search results

☐ * ☐ Procedure:based-on ☐ Procedure:encounter ☐ Procedure:instantiates-canonical ☐ Procedure:location

☐ Procedure:part-of ☐ Procedure:patient ☐ Procedure:performer ☐ Procedure:reason-reference ☐ Procedure:subject

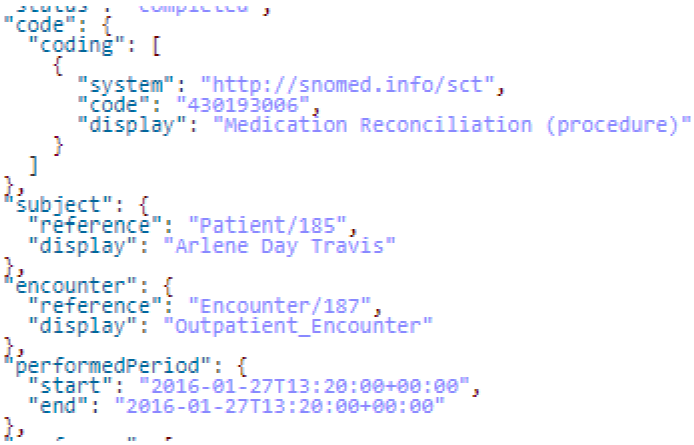
Sort Results

Sort By date Direction asc

Other Options

Limit 50

HEDIS TRC compliance can now be determined by comparing the medication reconciliation procedures returned by this query to the list of discharge documents returned by the previous one.



```
"status": "completed",
"code": {
  "coding": [
    {
      "system": "http://snomed.info/sct",
      "code": "430193006",
      "display": "Medication Reconciliation (procedure)"
    }
  ]
},
"subject": {
  "reference": "Patient/185",
  "display": "Arlene Day Travis"
},
"encounter": {
  "reference": "Encounter/187",
  "display": "Outpatient_Encounter"
},
"performedPeriod": {
  "start": "2016-01-27T13:20:00+00:00",
  "end": "2016-01-27T13:20:00+00:00"
},
```

Note that the patient’s displayed name in the subject field is not guaranteed to be unique, so comparisons should rely on the reference id instead.

Rest Client Guide

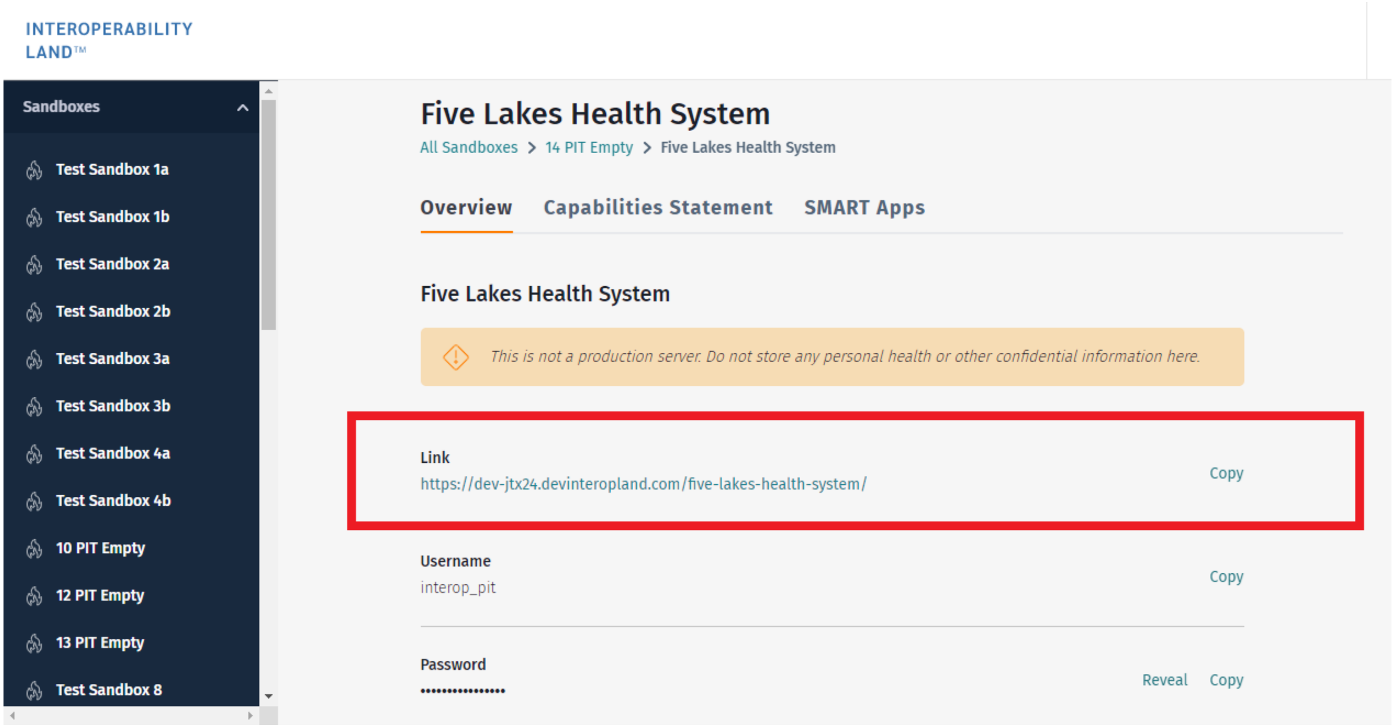
Performing a Basic Query in Postman

Postman is a tool for API testing that helps organize HTTP requests before sending them to a server. It can be used it as an alternative to the front end query tool.

First the URL needs to be found. The below is an example:

https://<<Interoperability Land PIT details>>.interopland.com/<<FHIR-PIT-NAME>>
/fhir/Patient?identifier=000003142

This URL and all other details needed for testing using Postman can be found in the Interoperability Land Pit Overview section for a given PIT. The Pit Tab Menu, covered earlier in this guide explains in detail where to find it. By copying the pit details on the Interoperability Land front end, a full version of the URL above can be obtained.



Five Lakes Health System

Five Lakes Health System

This is not a production server. Do not store any personal health or other confidential information here.

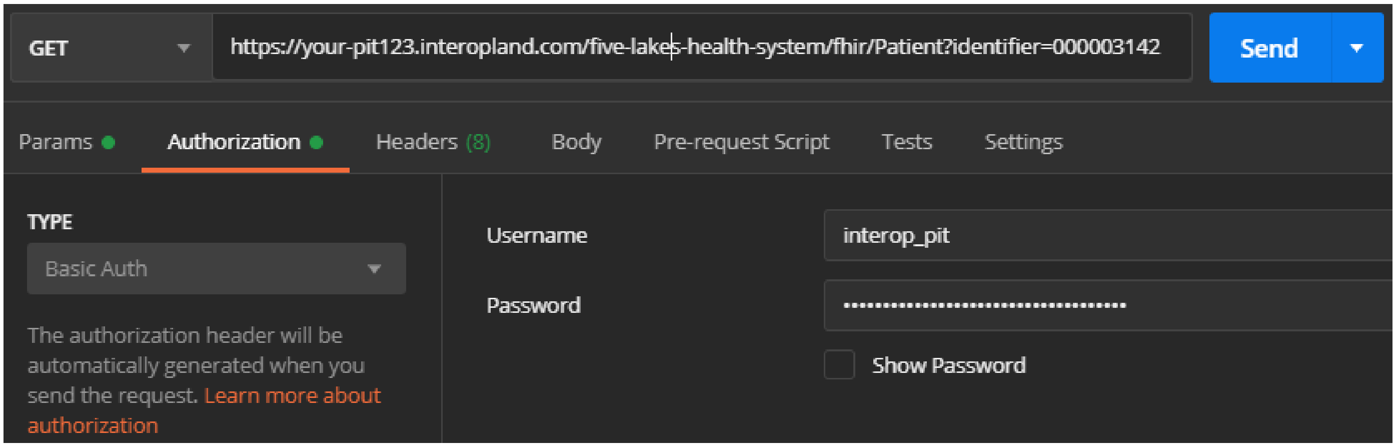
Link
https://dev-jtx24.devinteropland.com/five-lakes-health-system/

Username
interop_pit

Password
•••••••••••••••

Next, authorization needs to be set up in Postman. Further down in the above screenshot are fields for Username and Password. These go in the "Authorization" tab of Postman, with type set to 'Basic Auth'.

Once authorization has been set up, after clicking the 'Send' button in Postman, an HTTP GET request is generated and sent to the server to query for any **Patient** resources with an identifier(ssn) of 000003142.



https://your-pit123.interopland.com/five-lakes-health-system/fhir/Patient?identifier=000003142

TYPE: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Username: interop_pit

Password: ••••••••••••••••••••••••••••••••••••••

Utilizing Postman and the Restful API, it is possible to craft more complex queries. In the Appendix of this document, more advanced examples are given that show how to query for data over a range of dates as well as how to retrieve data from other FHIR resources related to patient.

Despite the versatility of the Restful API, use of external code libraries and APIs may be required in the cases where dynamic queries need to be built from the results of an initial query. The Appendix contains a small snippet generated from the HAPI FHIR java project that exemplifies a more complex querying scenario, and participants in the Connectathon should feel free to use their own preferred FHIR R4 libraries if they wish.

Appendix

Querying FHIR Resources: A Supplementary Guide to the Event Guide

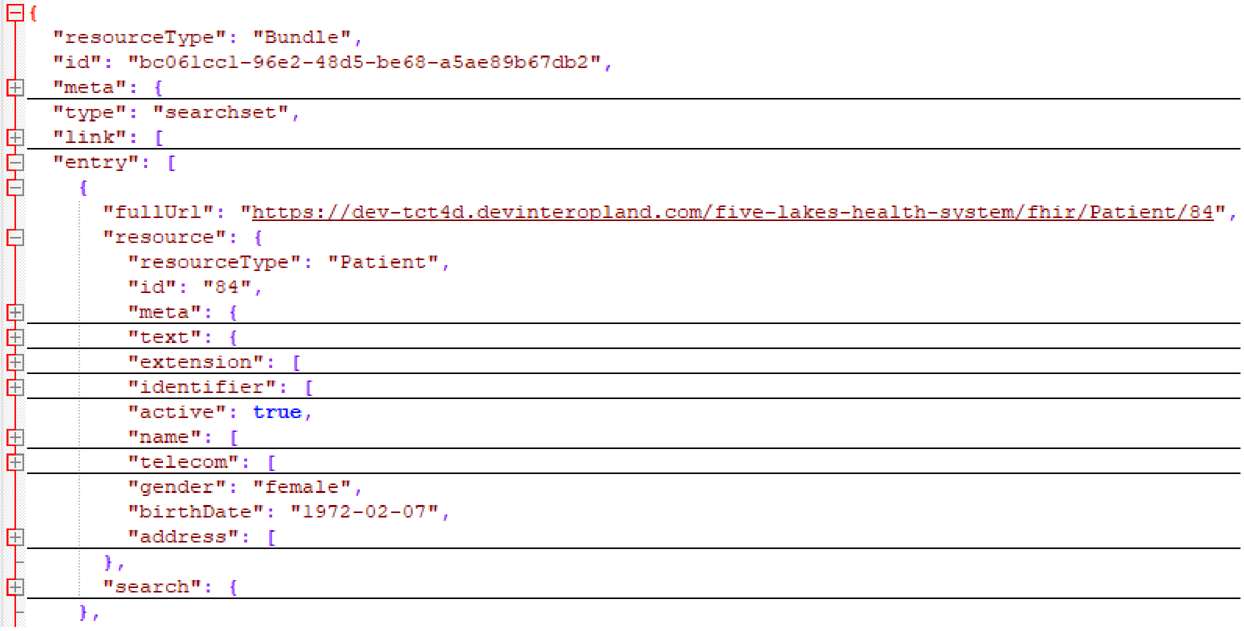
FHIR servers provide a number of ‘Resources’, or health-care related data objects that can be created, read, updated, or deleted. These resources have a variety of uses and have varied industry adoption. For the curious, a full list of resources and their level of maturity can be found here - **https://www.hl7.org/fhir/resourcelist.html**.

Resources can represent a variety of topics which may be useful to different organizations. For example, hospitals would likely need the **Patient** resource, as well as **Observations**, **Immunizations**, and **AllergyIntolerances**. Insurance companies may be more concerned with **Procedures**, **Claims**, and **Coverage** resources. University research labs might use the **ResearchSubject** or **SubstanceNucleicAcid** resource. There are also resources needed for server infrastructure, such as the **CapabilityStatement**, which is what the server uses to advertise what version and resources it knows about, and the **OperationDefinition**, which advertises what operations the server can perform on resources.

One useful way to work with FHIR resources is by utilizing ‘GET’ requests in Postman to query for the resources using the RESTful API.

In Postman, a GET request to:
https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient

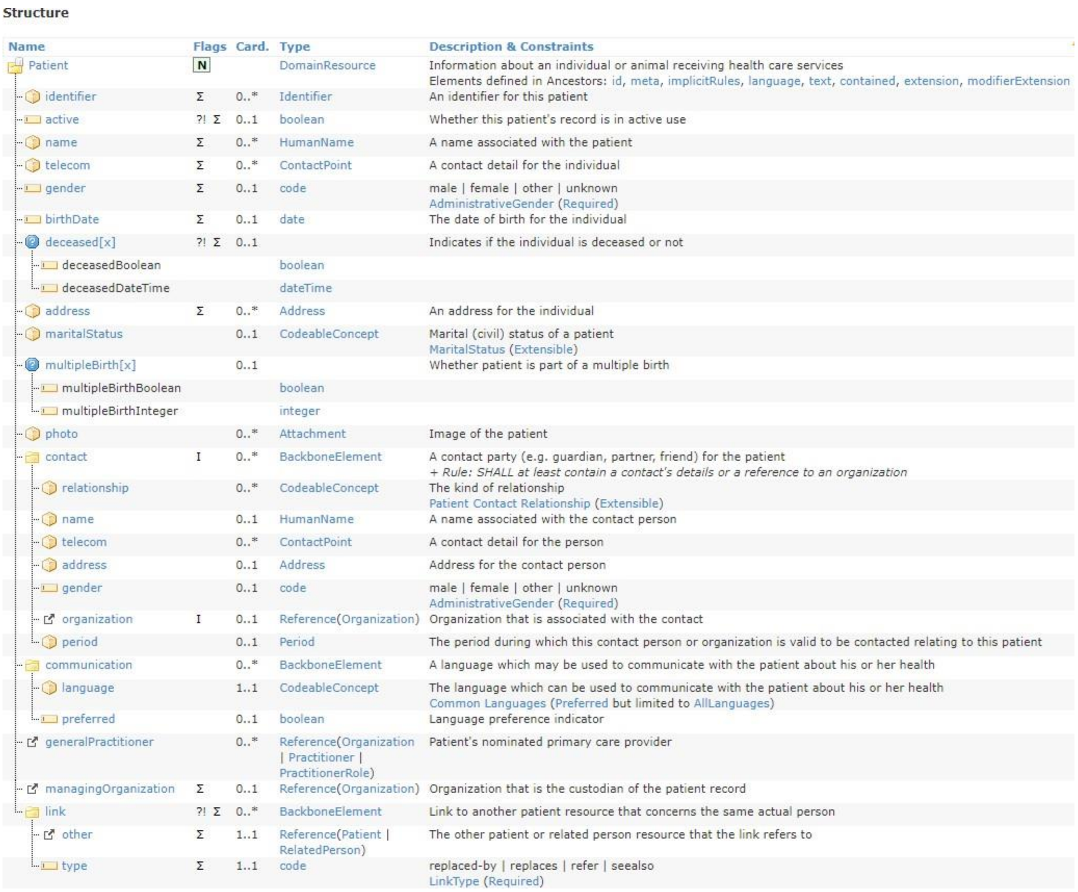
Will yield something like this:



```
{
  "resourceType": "Bundle",
  "id": "bc061cc1-96e2-48d5-be68-a5ae89b67db2",
  "meta": {
  "type": "searchset",
  "link": [
  "entry": [
    {
      "fullUrl": "https://dev-tct4d.devinteropland.com/five-lakes-health-system/fhir/Patient/84",
      "resource": {
        "resourceType": "Patient",
        "id": "84",
        "meta": {
        "text": {
        "extension": [
        "identifier": [
        "active": true,
        "name": [
        "telecom": [
        "gender": "female",
        "birthDate": "1972-02-07",
        "address": [
      },
      "search": {
    },
```

If it is difficult to read query responses due to all the data returned and displayed at once in Postman, the shortcut to fold results is Alt-0/Shift-Alt-0. This makes FHIR query responses easier to navigate and understand.

The fields in the above image match up roughly to the resource guide from HL7 FHIR listed below. The Σ flag in the below screenshot indicates that these fields will show up in a patient **summary** request, which is what the initial Postman request was for. Likewise, some fields are absent in the results – this is either because there's no data, or you need to request a non-summary version using a more focused request (like http://[url]/patient/[**id**]).



Structure

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal receiving health care services Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	An address for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient MaritalStatus (Extensible)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend) for the patient *+ Rule: SHALL at least contain a contact's details or a reference to an organization*
relationship		0..*	CodeableConcept	The kind of relationship Patient Contact Relationship (Extensible)
name		0..1	HumanName	A name associated with the contact person
telecom		0..*	ContactPoint	A contact detail for the person
address		0..1	Address	Address for the contact person
gender		0..1	code	male | female | other | unknown AdministrativeGender (Required)
organization	I	0..1	Reference(Organization)	Organization that is associated with the contact
period		0..1	Period	The period during which this contact person or organization is valid to be contacted relating to this patient
communication		0..*	BackboneElement	A language which may be used to communicate with the patient about his or her health
language		1..1	CodeableConcept	The language which can be used to communicate with the patient about his or her health Common Languages (Preferred but limited to AllLanguages)
preferred		0..1	boolean	Language preference indicator
generalPractitioner		0..*	Reference(Organization | Practitioner | PractitionerRole)	Patient's nominated primary care provider
managingOrganization	Σ	0..1	Reference(Organization)	Organization that is the custodian of the patient record
link	?! Σ	0..*	BackboneElement	Link to another patient resource that concerns the same actual person
other	Σ	1..1	Reference(Patient | RelatedPerson)	The other patient or related person resource that the link refers to
type	Σ	1..1	code	replaced-by | replaces | refer | seealso LinkType (Required)

Example query structure

[base URL] [organization name] [“/fhir/”] [resourceType]

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/DocumentReference

Returns all DocumentReference resources from the Five Lakes Health System PIT (default number of records is 20 and maximum records possible for return is 200)

Example of a Document Reference returned:

ResourceType	DocumentReference
Id	29106
masterIdentifier	`2deaf4a0-1b3a-4568-b129-35852ea49c2d`
Identifier	`2deaf4a0-1b3a-4568-b129-35852ea49c2d`
Description	`A03 of encounter Id = 2deaf4a0-1b3a-4568-b129-35852ea49c2d`
Subject.reference	Patient/424
Context.encounter.reference	`Encounter/29102`
Period.start	`2023-05-12T00:00:00+00:00`

Period.end	2023-05-12T00:00:00+00:00

Advanced query formatting

If the SSN for a Patient known, it is possible to look up any DocumentReference resources referencing that patient:

https://<<PIT DETAILS>>/new-hope-services/fhir/DocumentReference?subject:Patient.identifier=000002885

The ‘000002885’ can be replaced with other SSN to find a different patient’s DocumentReference resources.

ResourceType	DocumentReference
Id	29106
masterIdentifier	2deaf4a0-1b3a-4568-b129-35852ea49c2d
Identifier	2deaf4a0-1b3a-4568-b129-35852ea49c2d
Description	A03 of encounter Id = 2deaf4a0-1b3a-4568-b129-35852ea49c2d
Subject.reference	Patient/424
Context.encounter.reference	Encounter/29102
Period.start	2023-05-12T00:00:00+00:00
Period.end	2023-05-12T00:00:00+00:00

Querying for a range of values

The below example searches for any **Encounter** resources within a date range, with criteria: “greater than or equal to 2025-01-01 and less than or equal to 2025-01-31”.

- https://<< PIT DETAILS>>.interopland.com/five-lakes-health-system /fhir/Encounter?date=ge2025-01-01&date=le2025-01- 31&patient=170

The next example searches for patients linked to a **Procedure** having Snomed code 430193006 (a medication reconciliation)

- https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/ fhir/Patient?_has:Procedure:subject:code=430193006

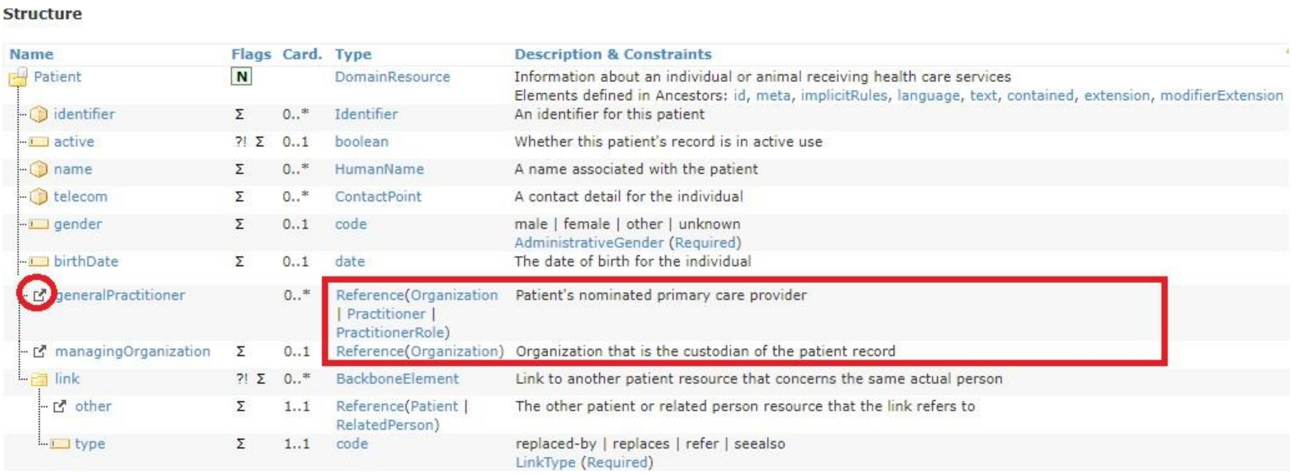
Including additional resources can be done with use of the **_include** or **_revinclude** query parameters. These can be used when the base resource (in this case, **Patient**) has a link to an external resource, or if the external resource (**Encounter**), links back to it.

- https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/ /fhir/Patient?identifier=000002875&_count=1&_revinclude=Encounter:subject

Data linkage

It may be useful to conceptualize the data as a SQL table, despite the back-end technology being No-SQL. In doing this, the concept of Foreign Keys can be used to navigate around and discover data relationships.

The equivalent mechanism is in place for IOL data as well, in the form of References. These are incredibly valuable when crafting more complex queries, such as the those seen in the next section. These are best discovered through the individual resource's documentation on the Hl7 FHIR site, and searching for 'Reference' under the type column - https://www.hl7.org/fhir/patient.html



Structure

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal receiving health care services Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
generalPractitioner		0..*	Reference(Organization | Practitioner | PractitionerRole)	Patient's nominated primary care provider
managingOrganization	Σ	0..1	Reference(Organization)	Organization that is the custodian of the patient record
link	?! Σ	0..*	BackboneElement	Link to another patient resource that concerns the same actual person
other	Σ	1..1	Reference(Patient | RelatedPerson)	The other patient or related person resource that the link refers to
type	Σ	1..1	code	replaced-by | replaces | refer | seealso LinkType (Required)

An alternative way to identify resource references using the HL7 FHIR site is by looking for a block just above the “Resource Content’ section. The block states: ‘This resource is referenced by:. ’, followed by a list of resources.

For reference, the current HL7 R4 paragraph lists all of these resources that link to Patient:
Annotation, Signature, Account, AdverseEvent, AllergyIntolerance, Appointment, AppointmentResponse, AuditEvent, Basic, BiologicallyDerivedProduct, BodyStructure, CarePlan, CareTeam, ChargeItem, Claim, ClaimResponse, ClinicalImpression, Communication, CommunicationRequest, Composition, Condition, Consent, Contract, Coverage, CoverageEligibilityRequest, CoverageEligibilityResponse, DetectedIssue, Device, DeviceRequest, DeviceUseStatement, DiagnosticReport, DocumentManifest, DocumentReference, Encounter, EnrollmentRequest, EpisodeOfCare, ExplanationOfBenefit, FamilyMemberHistory, Flag, Goal, Group, GuidanceResponse, ImagingStudy, Immunization, ImmunizationEvaluation, ImmunizationRecommendation, Invoice, List, MeasureReport, Media, MedicationAdministration, MedicationDispense, MedicationRequest, MedicationStatement, MolecularSequence, NutritionOrder, Observation, itself, Person, Procedure, Provenance, QuestionnaireResponse, RelatedPerson, RequestGroup, and others.

In theory, you can get a very complete picture of a Patient resource in one json request by using **_revinclude** on the that whole list. In practice, not every system will implement all resources, or security features may be in place, so your results may vary. Note that overzealous usage of **_revincludes** may stress the server, so judicious usage is advised.

For an example, examine the parameters in the following query:

https://<<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient?identifier=000002867&_count=1&_revinclude=DocumentReference:subject&_revinclude=Encounter:subject

You’ll notice the query contains the following details:

- **_count**=1
  - Limits our resultset to only one patient, to save on server load.
- **identifier**=000002867
  - Patient resource has SSN as identifier, so it can be searched upon
- **_revinclude**=DocumentReference:subject
  - (Fetches and includes DocumentReferences that reference the patient)
- **_revinclude**=Encounter:subject
  - (Fetches and includes Encounters that reference the patient)

Crafting more complex queries to answer questions

The FHIR server offers a moderately powerful querying functionality accessible through endpoints used in earlier exercises. It's capable of some simple join operations, as well as simple filtering. If you are familiar with SQL, the mechanisms are similar to the JOIN and WHERE statements albeit with limitations.

Here's a basic summary query for 20 patients:

https:// <<PIT DETAILS>>.interopland.com/five-lakes-health-system/fhir/Patient

And below is a more complex query for 20 patients that have **DocumentReferences** that both link back to the patient _and_ have A01 in the description. Note that the query does **NOT** include the **DocumentReferences** in the result.

https:// <<PIT DETAILS>>.interopland.com/five-lakes-health-system
/fhir/Patient?_has:DocumentReference:patient:description=A01

The **_has** parameter utilizes references on the **DocumentReference** that point to patient (as seen in the last section).

Similar functionality can be achieved with **_revinclude**, but this would include the **DocumentReference** in the json results. (Of note, **_has** can also achieve this if **_contained=true** is in the query**)**

Limitations to out-of-the-box queries

- No sub-queries without customizing the server
- (i.e.: can't query resultset from a complex query)
- No programmatically extended ranges
- No way to dynamically query 'between 1/1/2020 and 30 days from then

Workarounds to these limitations

- FHIR client libraries implementing the desired resources for your desired standard version (currently: **R4**)
- Available in java, python, .net
- Use lists/arrays to hold the results, then foreach over them to filter/process or programmatically perform follow-up queries against FHIR server as needed.

```
public static void main(String[] args) {
    FhirContext ctx = FhirContext.forR4();
    String serverBase = "https://replace-this-with-your-details.interopland.com/five-
lakes-health- system/fhir/";

    IGenericClient gclient = ctx.newRestfulGenericClient(serverBase);
    gclient.registerInterceptor(
        new BasicAuthInterceptor("interop_pit", "your-password-goes-here"));

    class userDischargeDate {
      String userId;
      String ADT30DayRangeStart;
      String ADT03DischargeEndDate;
      boolean followupEncounter = false;

      public userDischargeDate(String inUserId, Date ADT03DischargeDate) {
        userId = inUserId;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        ADT30DayRangeStart = sdf.format(ADT03DischargeDate);
        ADT03DischargeEndDate = sdf
            .format(DateUtils.addDays(ADT03DischargeDate, 30));
      }

    }

    List<userDischargeDate> Adt03sWithUsers = new ArrayList<userDischargeDate>();
    List<String> followUpEncountersWithin30Days;

    Bundle ADT03Results = gclient.search().forResource(DocumentReference.class)
        .where(DocumentReference.DESCRIPTION.contains().value("A03"))
        .returnBundle(Bundle.class).execute();

    ADT03Results.getEntry().forEach((entry) -> {
      DocumentReference docRef = (DocumentReference) entry.getResource();
      if (docRef.hasSubject()) {
        Reference subject = docRef.getSubject();
        Adt03sWithUsers
            .add(new userDischargeDate(subject.getId(), docRef.getDate()));
      }
    });
    Adt03sWithUsers.forEach((userDischargeDate) -> {
      Bundle EncounterResults = gclient.search().forResource(Encounter.class)
          .where(Encounter.PATIENT.hasId(userDischargeDate.userId))
          .and(Encounter.DATE.afterOrEquals()
              .day(userDischargeDate.ADT30DayRangeStart))
          .and(Encounter.DATE.beforeOrEquals()
              .day(userDischargeDate.ADT30DayRangeStart))
          .returnBundle(Bundle.class).execute();



      if (EncounterResults.hasEntry()) {
        userDischargeDate.followupEncounter = true;
        System.out.println("Patient: " + userDischargeDate.userId
            + " had a follow-up visit within 30 days");}
```